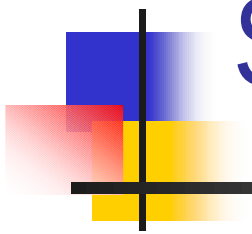


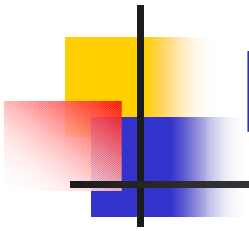
On the Value Locality of Store Instructions

Kevin M. Lepak

Mikko H. Lipasti

University of Wisconsin—Madison

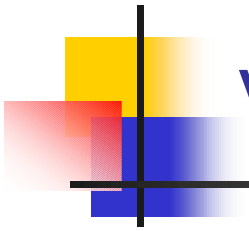
http://www.ece.wisc.edu/~pharm



Motivation

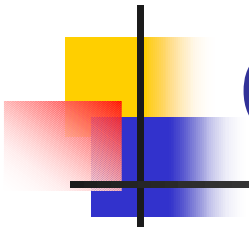
- Value Locality (VL) is Real
  - More than 30 VL/VP papers
  - Patents granted for VL work (AMD, Gabbay)
- VL has been traditionally used to speed up the next state function (the “means” of computing)
- VL has not been explored (generally) for the output function (the “end” of computing)

Is there any benefit to exploiting VL for outputs?



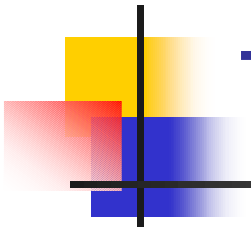
VL of Memory Writes—Who Cares?

- Reduction of writebacks/dirty lines is desirable
  - Less data traffic
  - Fewer invalidates
  - Reduced pressure on WB buffers
- Writes comparatively expensive
  - Requires multi-porting/banking circuit tricks
- Removal of “unnecessary” value storage seems intuitively satisfying
  - Do unnecessary writes imply unnecessary computation?



Outline

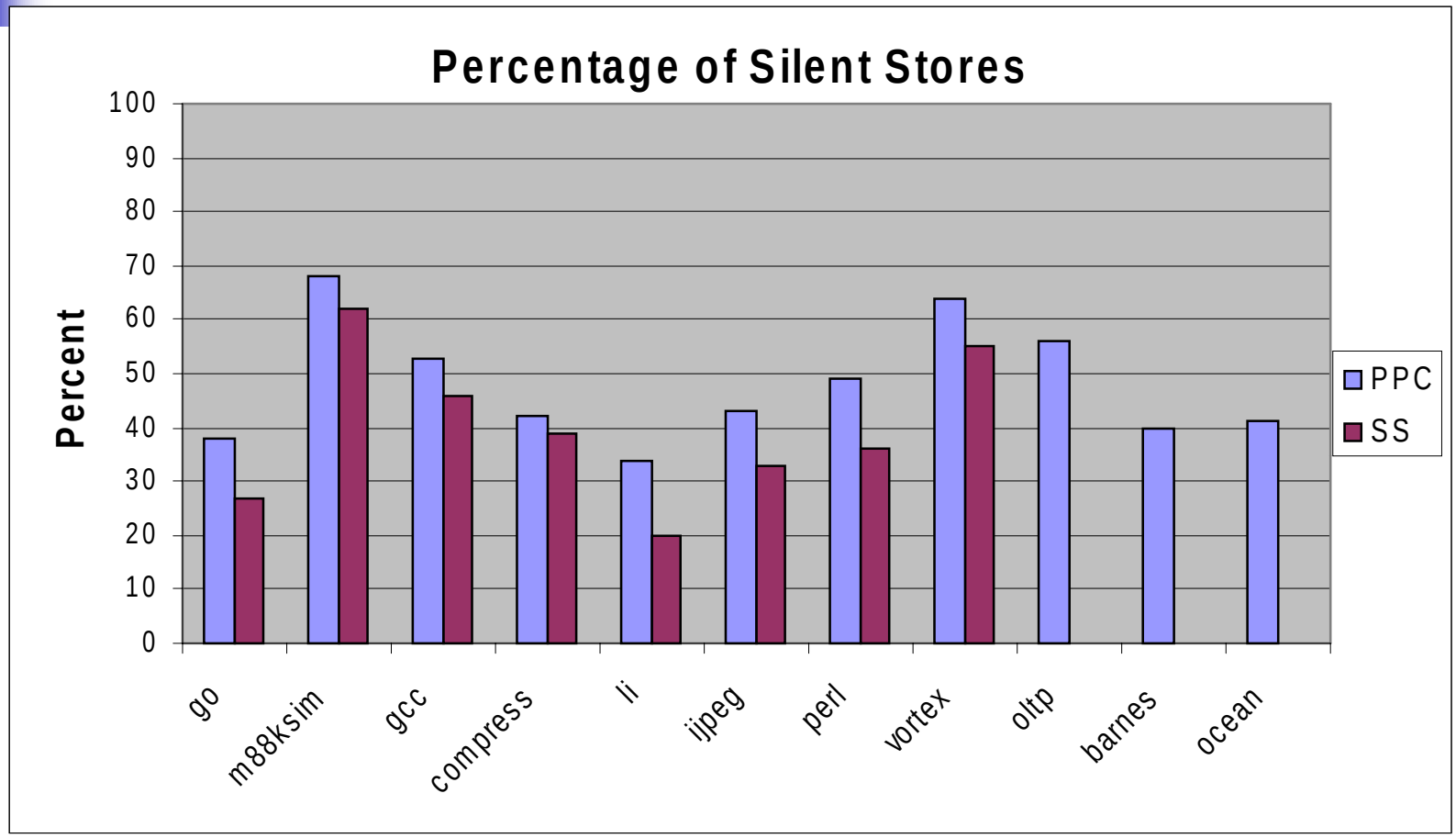
- Motivation
- **Introduction to Silent Stores**
- Uniprocessor Results
- Multiprocessor True/False Sharing
- New Definition of False Sharing
- Multi-Processor Results
- Conclusions



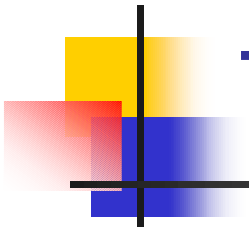
Terminology

- ***Silent Store:*** A memory write that does not change the system state

Silent Stores—Is this for Real?

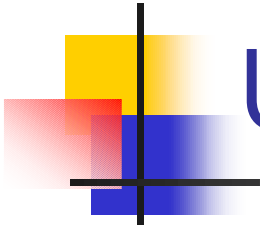


Percentage of silent stores is non-trivial in all cases, 20%-68%



Terminology

- ***Silent Store:*** A memory write that does not change the system state
- ***Store Verify:*** A load, compare, and subsequent store (if non-silent) operation
- ***Store Squashing:*** Removal of a silent store from program execution
  - Dynamically
  - Statically

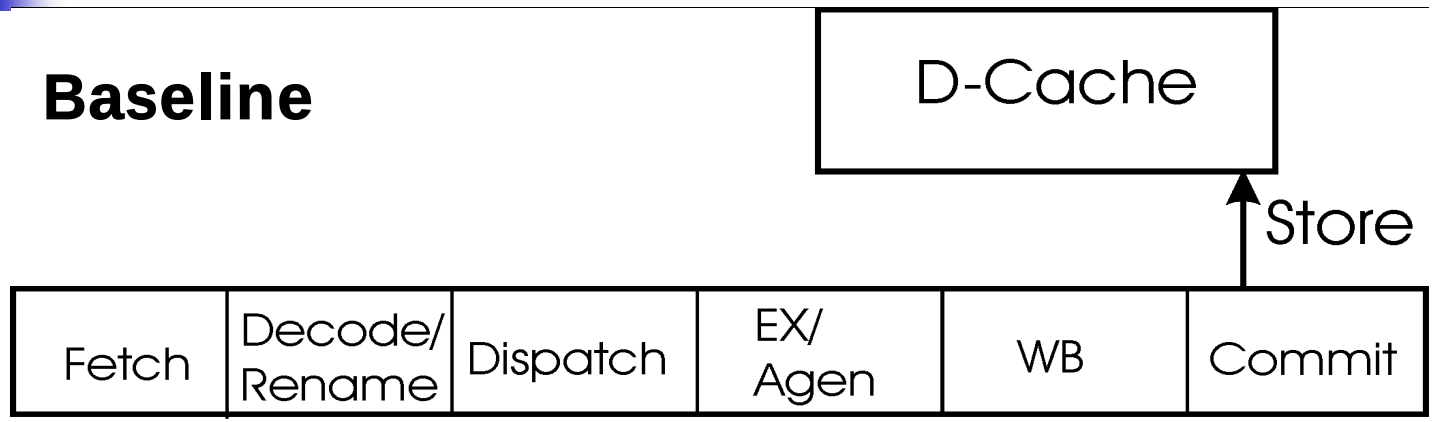


Uniprocessor Machine Model

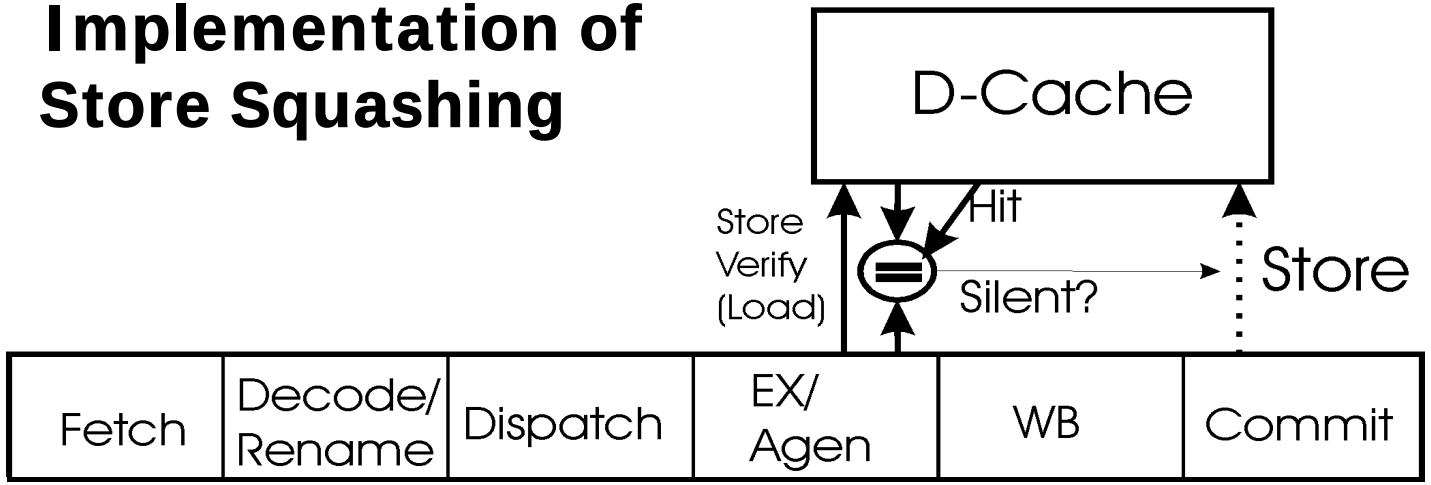
- SimpleScalar simulator
  - 64 entry RUU; 8 issue
  - 64K Gshare branch predictor
  - 64K each split I/D cache
  - 1MB L2 unified cache
  - 16 entry load/store queue
  - 4 memory load ports; 1 store port (4-wide version of the 2-wide 21164)

Squashing Mechanism

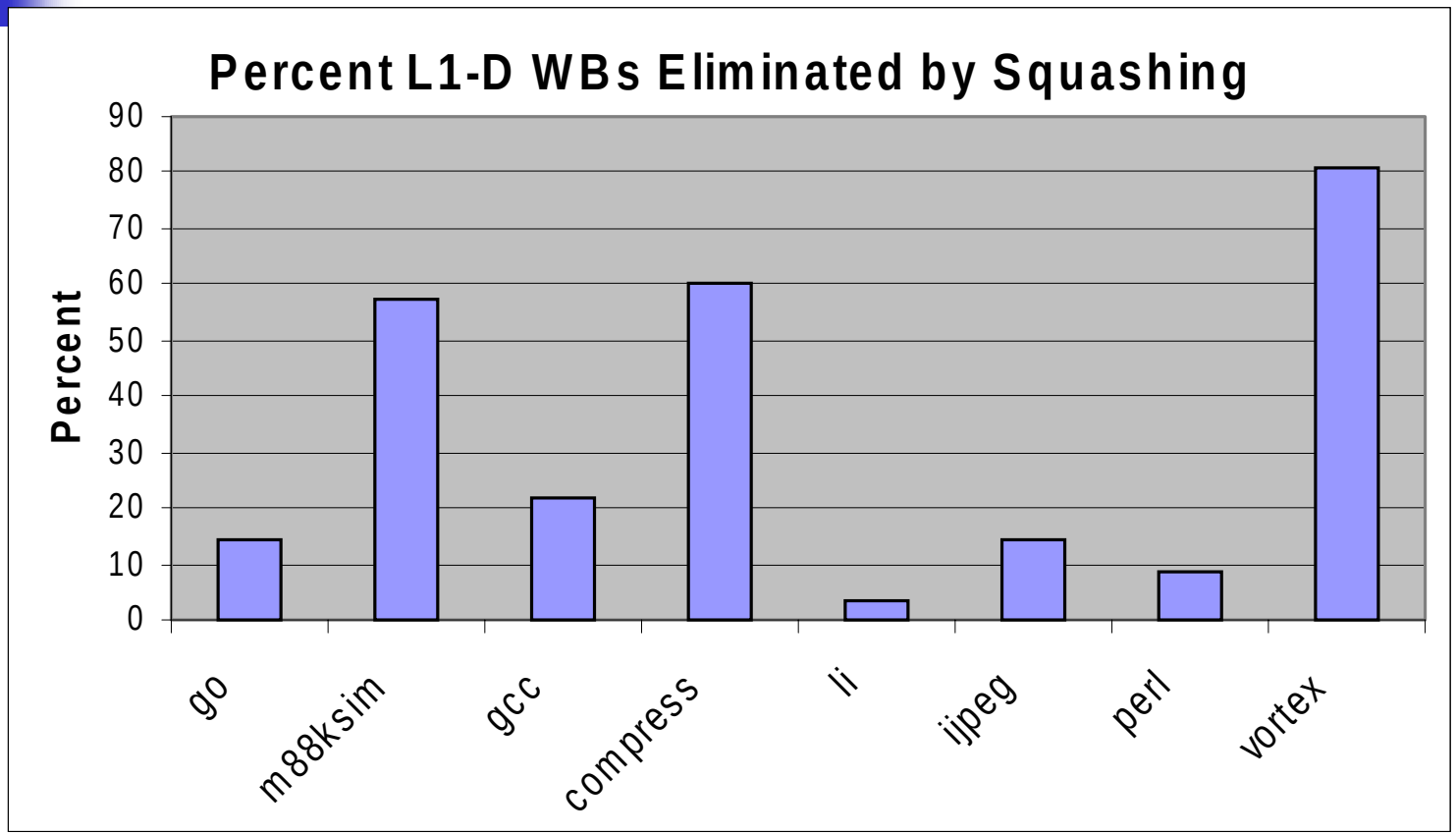
Baseline



Implementation of Store Squashing

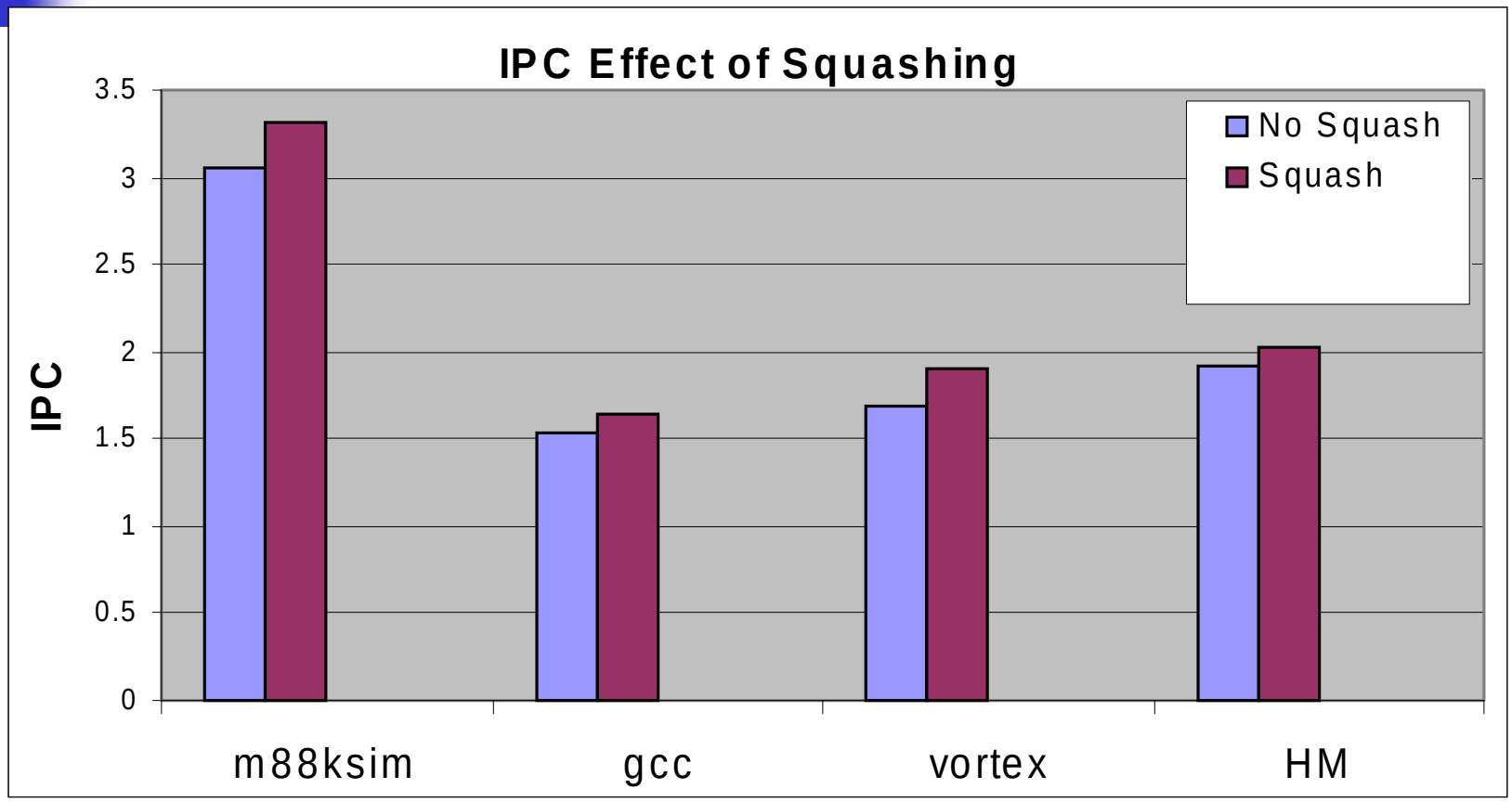


Writebacks Eliminated



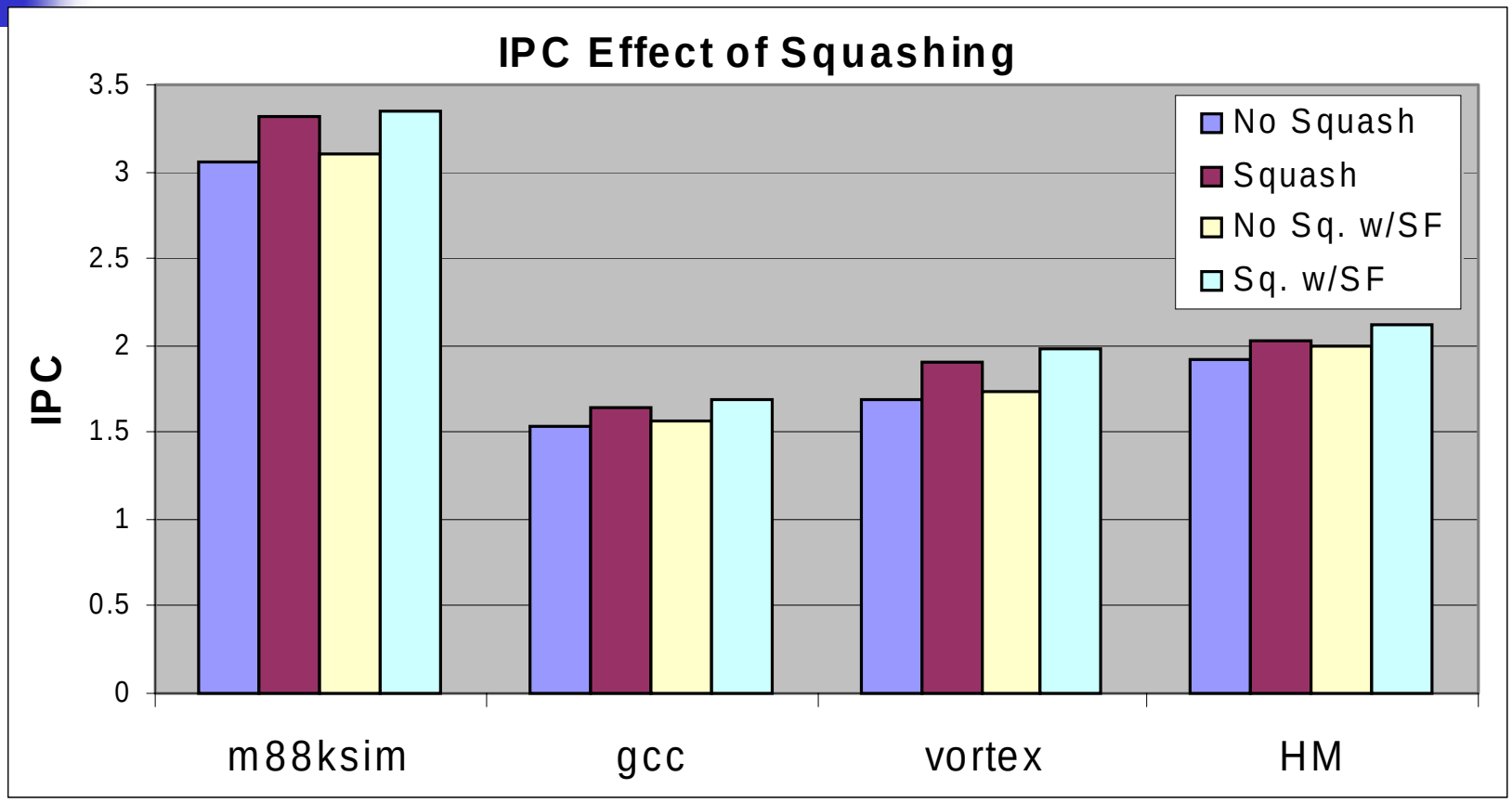
Substantial WB elimination by simplistic store verify/squash (14%-81% for cases with non-trivial WBs in the baseline case)

IPC Effects

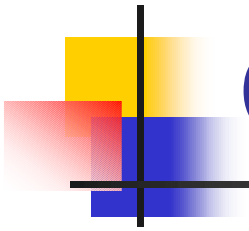


7.6%, 7.9%, 14%, and 6.3% speedup of squashing over no squashing for m88ksim, gcc, vortex, and HM

IPC Effects

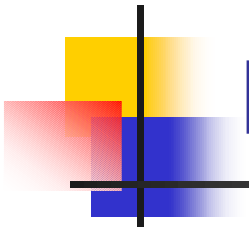


Squashing provides more benefit than store forwarding



Outline

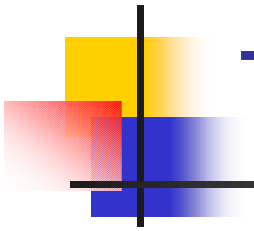
- Motivation
- Introduction to Silent Stores
- Uniprocessor Results
- **Multiprocessor True/False Sharing**
- New Definition of False Sharing
- Multi-Processor Results
- Conclusions



Multiprocessor True/False Sharing

- Dubois et. al: ISCA-1993
  - Address-based definition
  - Considers all “side-effects” (non-critical words) brought in by a cache miss and future accesses to the line

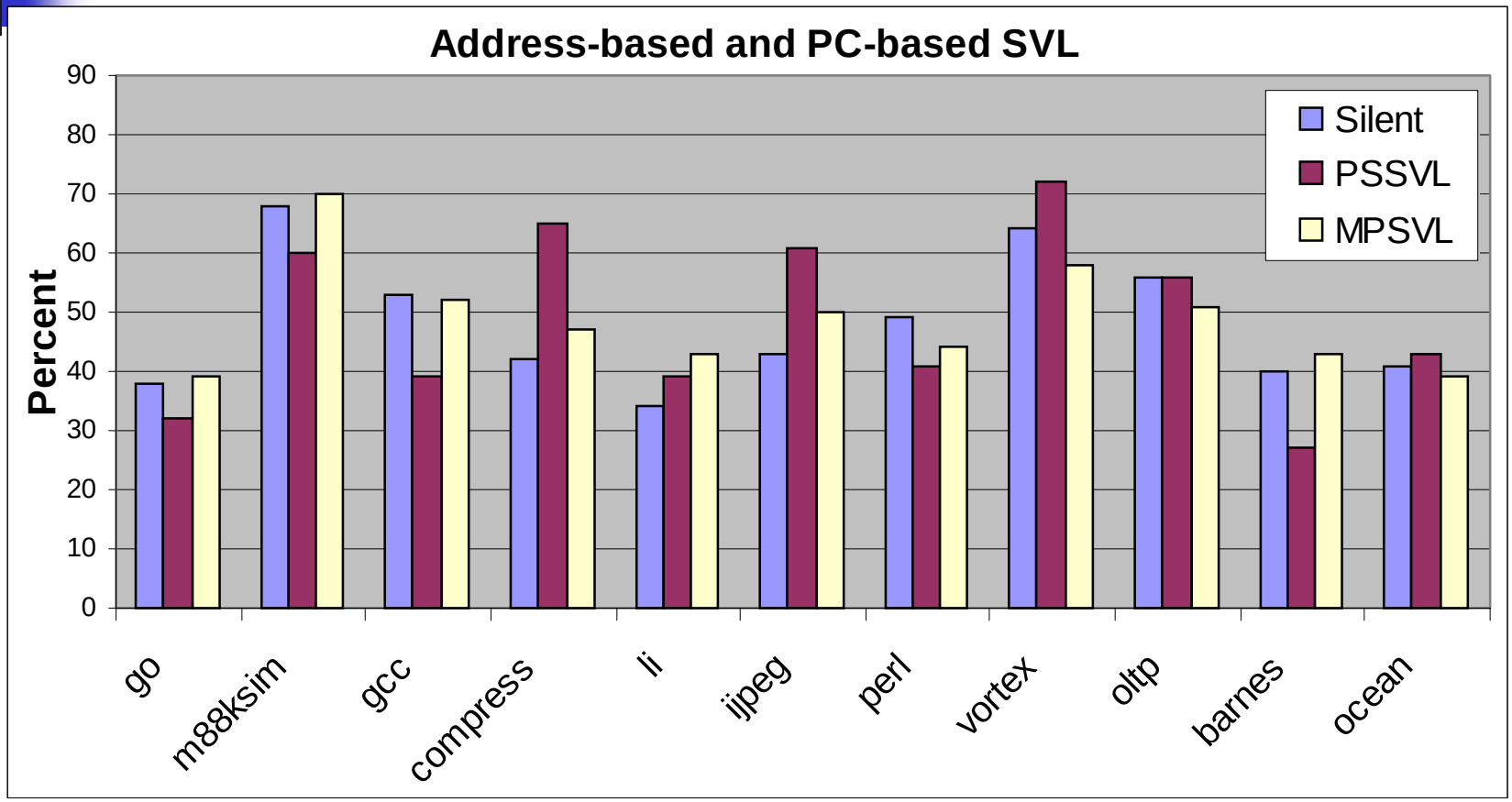
Does not consider silent stores or data value prediction



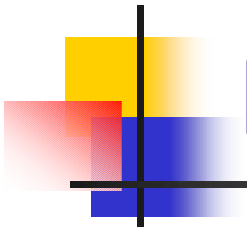
Terminology

- ***Program Structure Store Value Locality (PSSVL):*** The value locality exhibited by a given static store (can write to many addresses)
- ***Message Passing Store Value Locality (MPSVL):*** The value locality exhibited for a specific memory location (can be written by many PCs)
- ***Stochastically Silent Store:*** A store value which is trivially predictable by any well known method

MPSVL and PSSVL



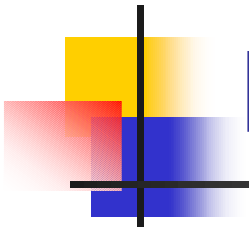
Percentage of stochastically silent (PSSVL, MPSVL) stores is non-trivial
27%-72% for PSSVL, 39%-70% for MPSVL



New Definition of False Sharing

Extend Dubois' definition with store value locality:

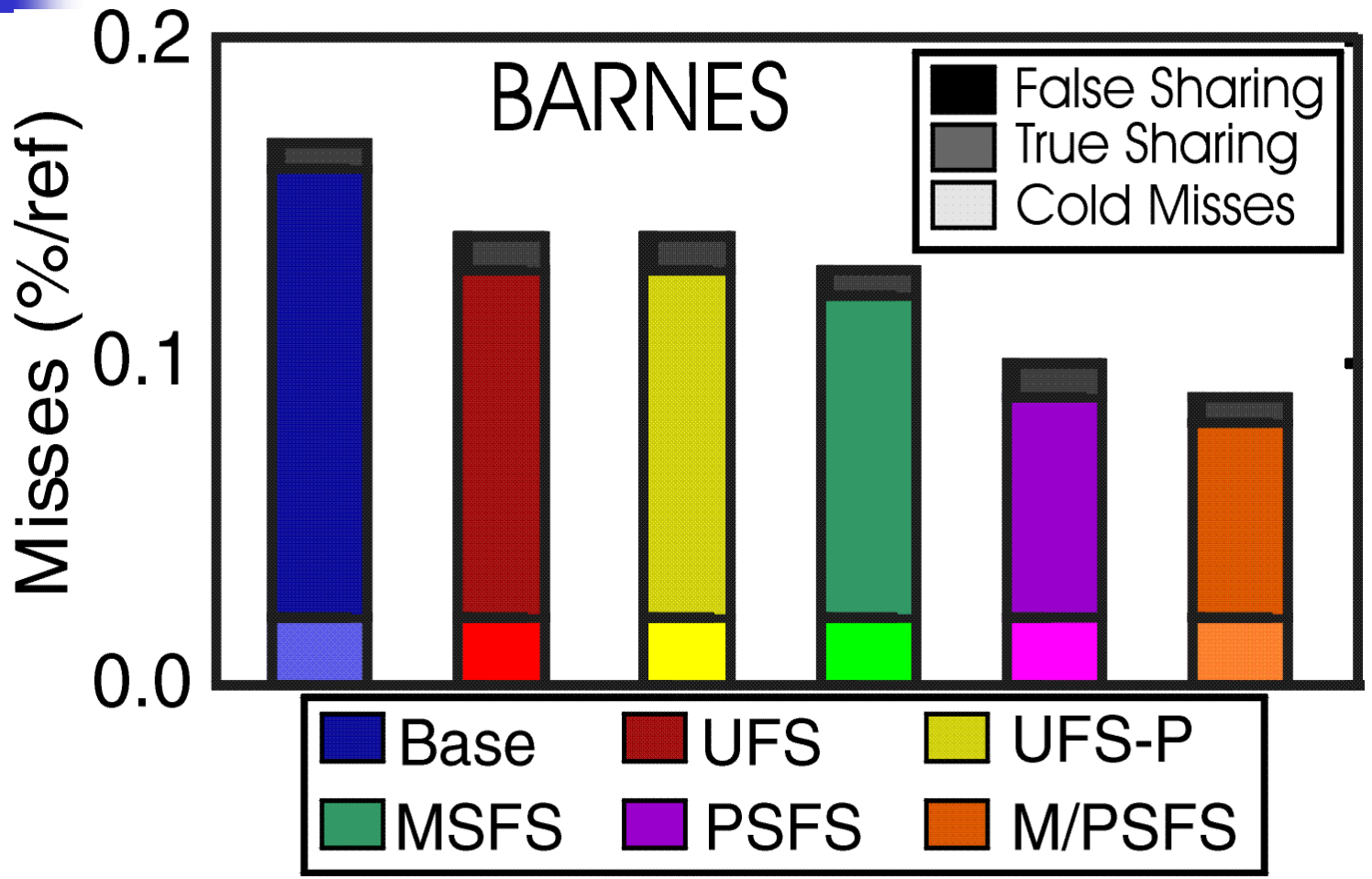
- ***Update False Sharing (UFS)***
  - Consider (update) silent stores
- ***Stochastic False Sharing (SFS)***
  - Consider stochastically (predictable by any well known method) silent stores
  - ***Message-passing Stochastic False Sharing (MSFS):***
    - Exploit value locality based on effective memory address
  - ***Program-structure Stochastic False Sharing (PSFS):***
    - Exploit value locality based on instruction address (PC)



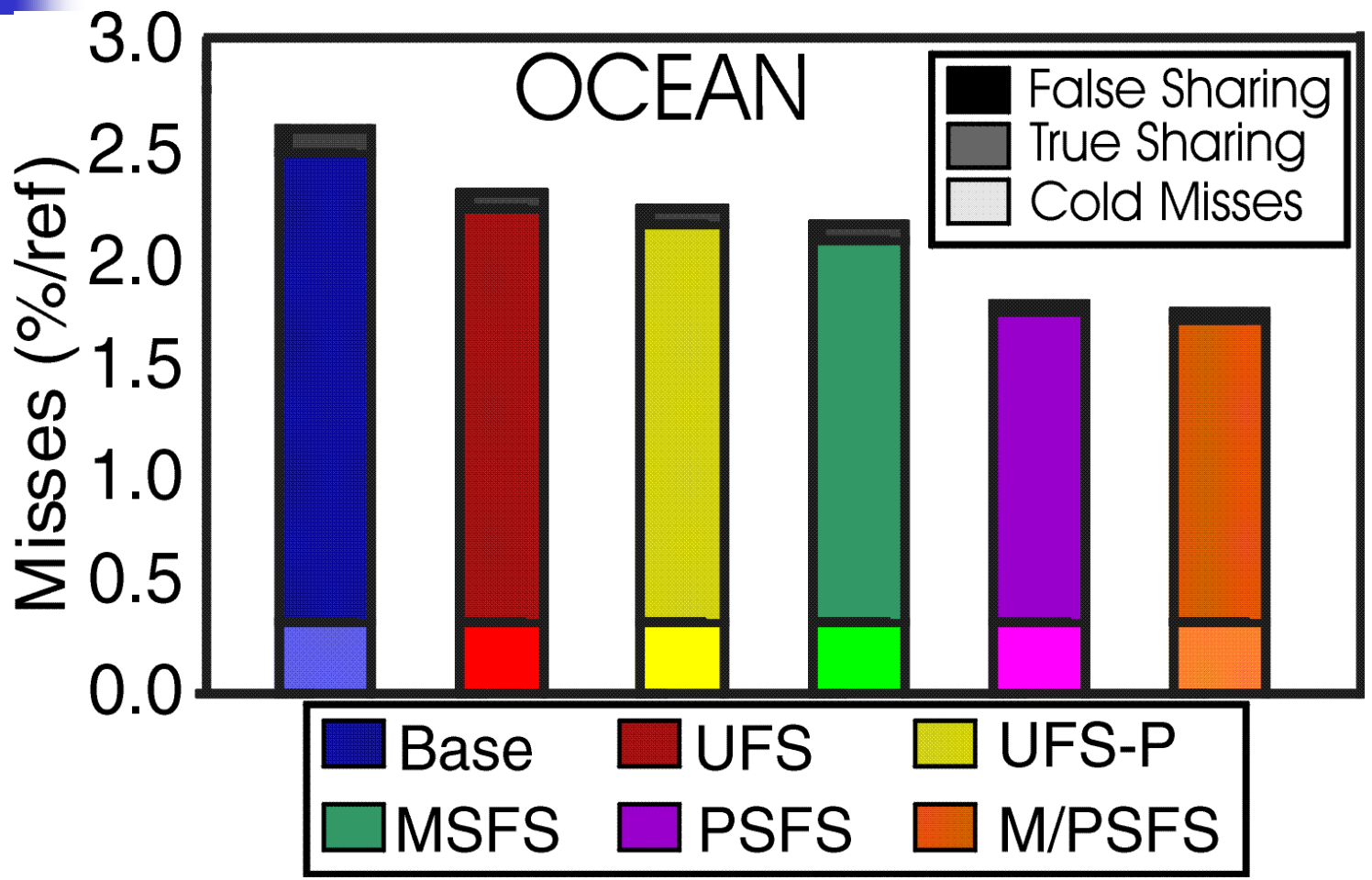
Machine Model

- SimOS-PPC full system simulator
  - 4 processors
  - 1MB data cache
  - 4K direct-mapped stride predictor for Program-structure and Message-passing store value locality
- Remove all silent and stochastically silent stores from the cache hierarchy
  - Limit study—must have a mechanism to exploit (subject of current research)

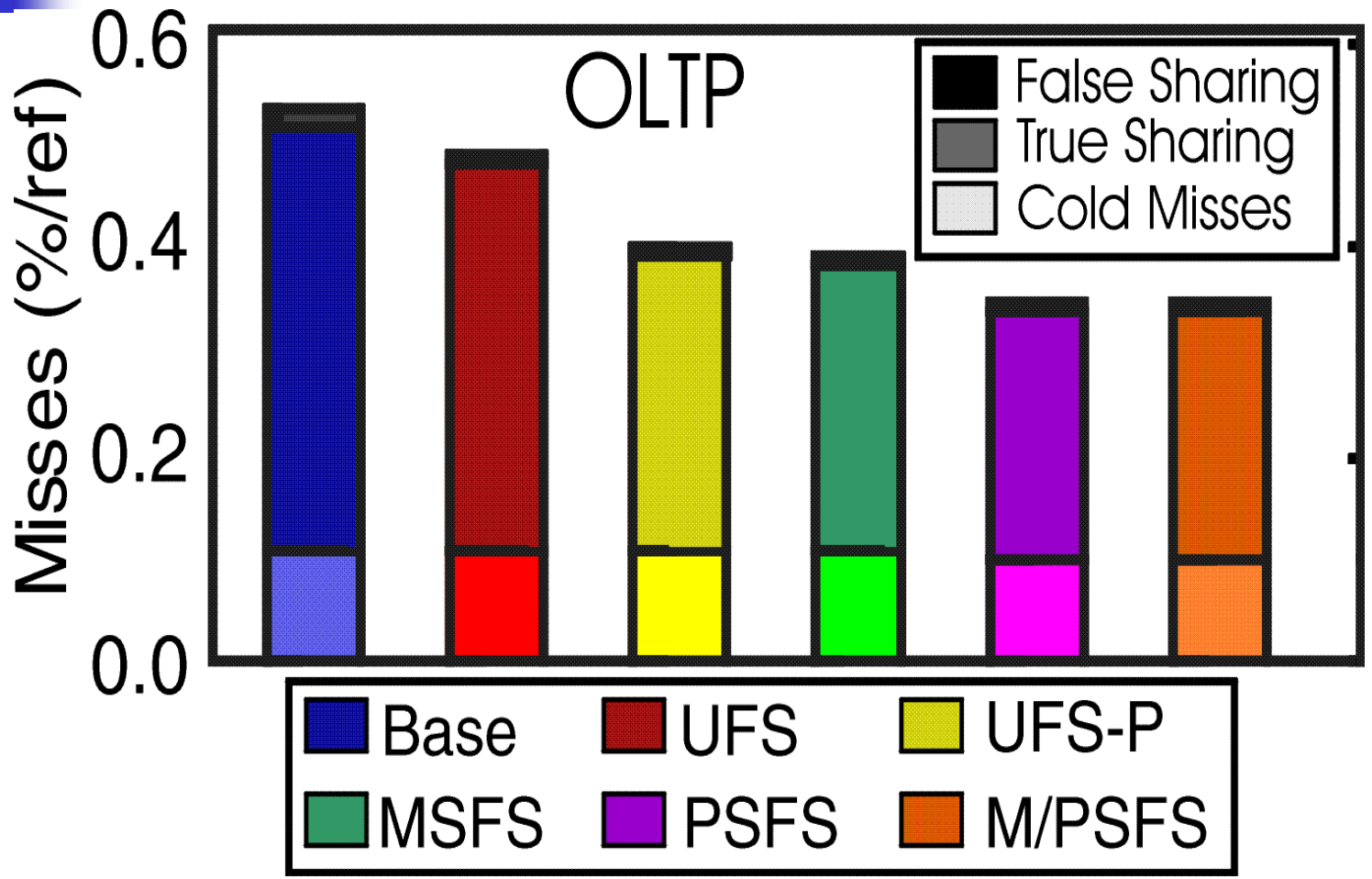
Multiprocessor Sharing

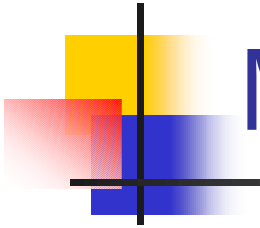


Multiprocessor Sharing



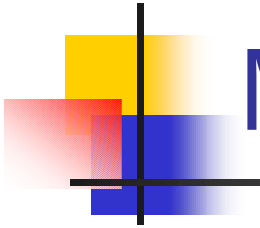
Multiprocessor Sharing





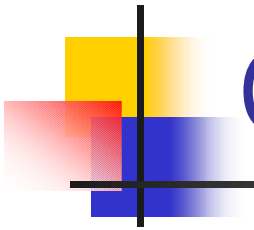
Multiprocessor Sharing

- Measurable reduction in true/false sharing for simple update silent squashing (UFS)
- Substantial reductions by squashing update silent store hits and misses (UFS-P) and stochastically silent stores (SFS)
- Squashing store misses (UFS-P) can be substantially better than simple UFS
  - Motivates silence confidence mechanism for store misses



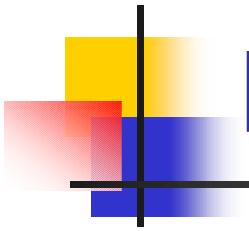
Multiprocessor Traffic

- Measurable reduction in invalidate traffic for simple update silent store squashing (UFS)—more effective than Exclusive state
  - Substantial reduction for UFS-P and Stochastic False Sharing (SFS)
- Writeback data traffic reduction by squashing update silent store hits and misses (UFS-P)
  - 5%-82% in *oltp*
  - 16%-17% in *ocean*
  - 5%-16% in *barnes*



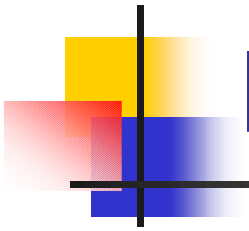
Conclusions

- Significant store value locality exists
  - MPSVL (includes update silent stores)
  - PSSVL
- Uniprocessor performance can be enhanced by squashing silent stores
- A new definition of sharing is given which accounts for update/stochastically silent stores
- We can exploit the new sharing definitions to reduce address and data bus traffic
  - UFS: Implementation given, non-trivial results
  - SFS: Limit study shows significant potential



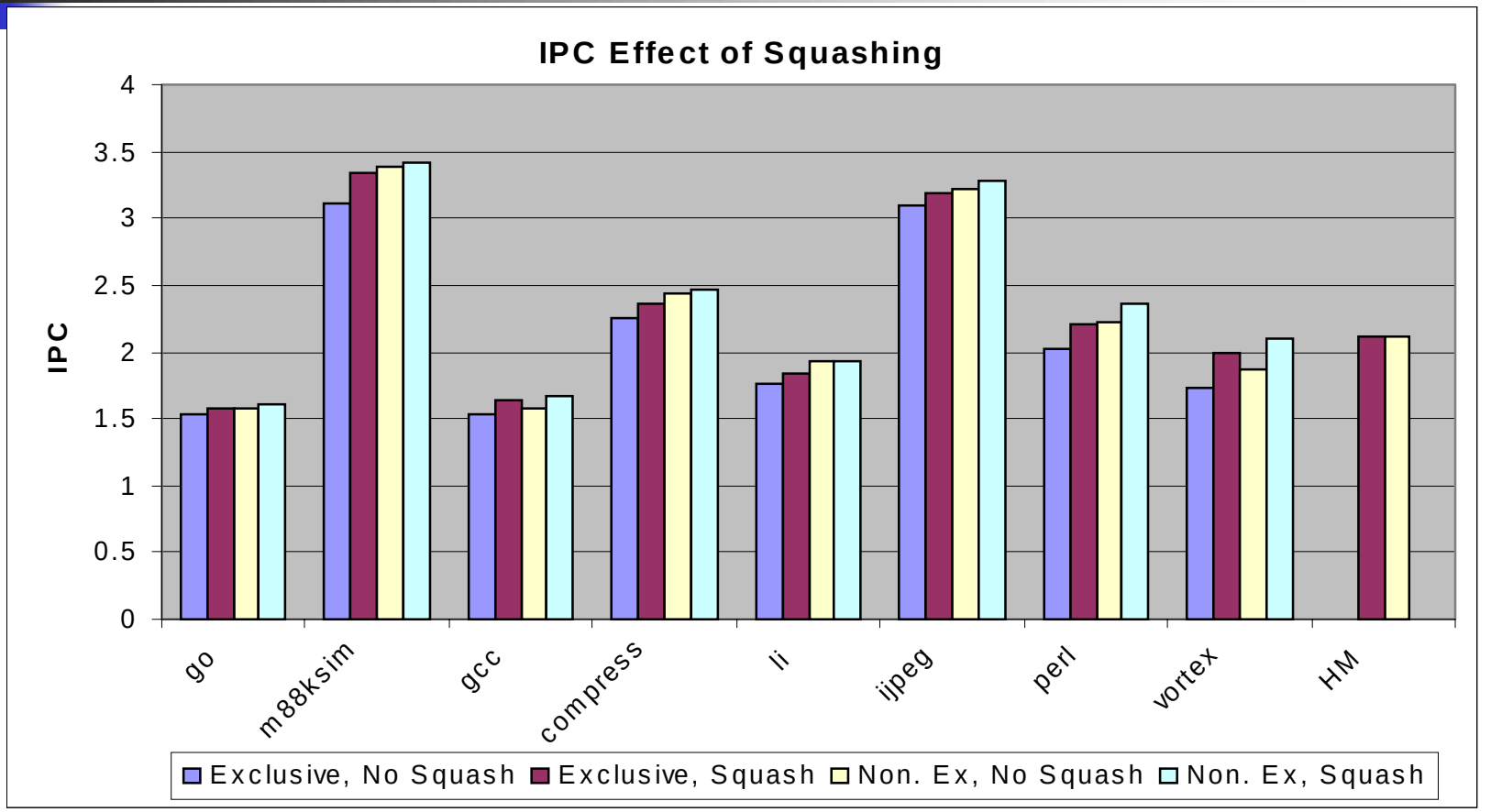
Future Work

- Characterization of silent stores
- Silence prediction and confidence mechanisms
- Implementation of SFS mechanism
- . . .

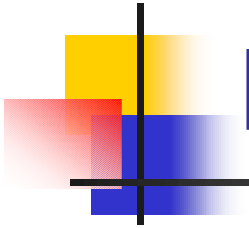


Backup Slides

IPC—Continued



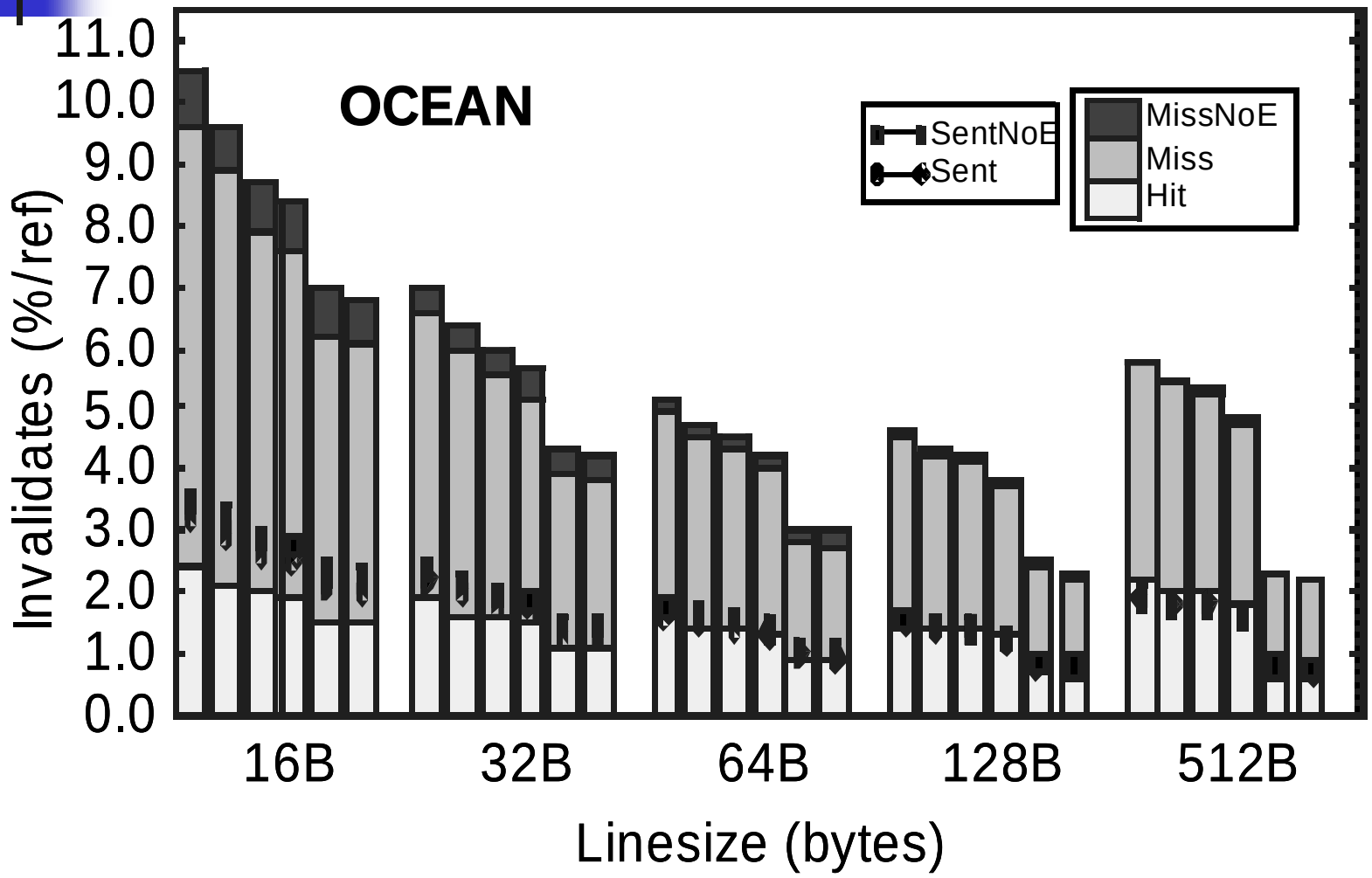
Squashing in an exclusive 4L/1S memory system equivalent to non-exclusive/no squash



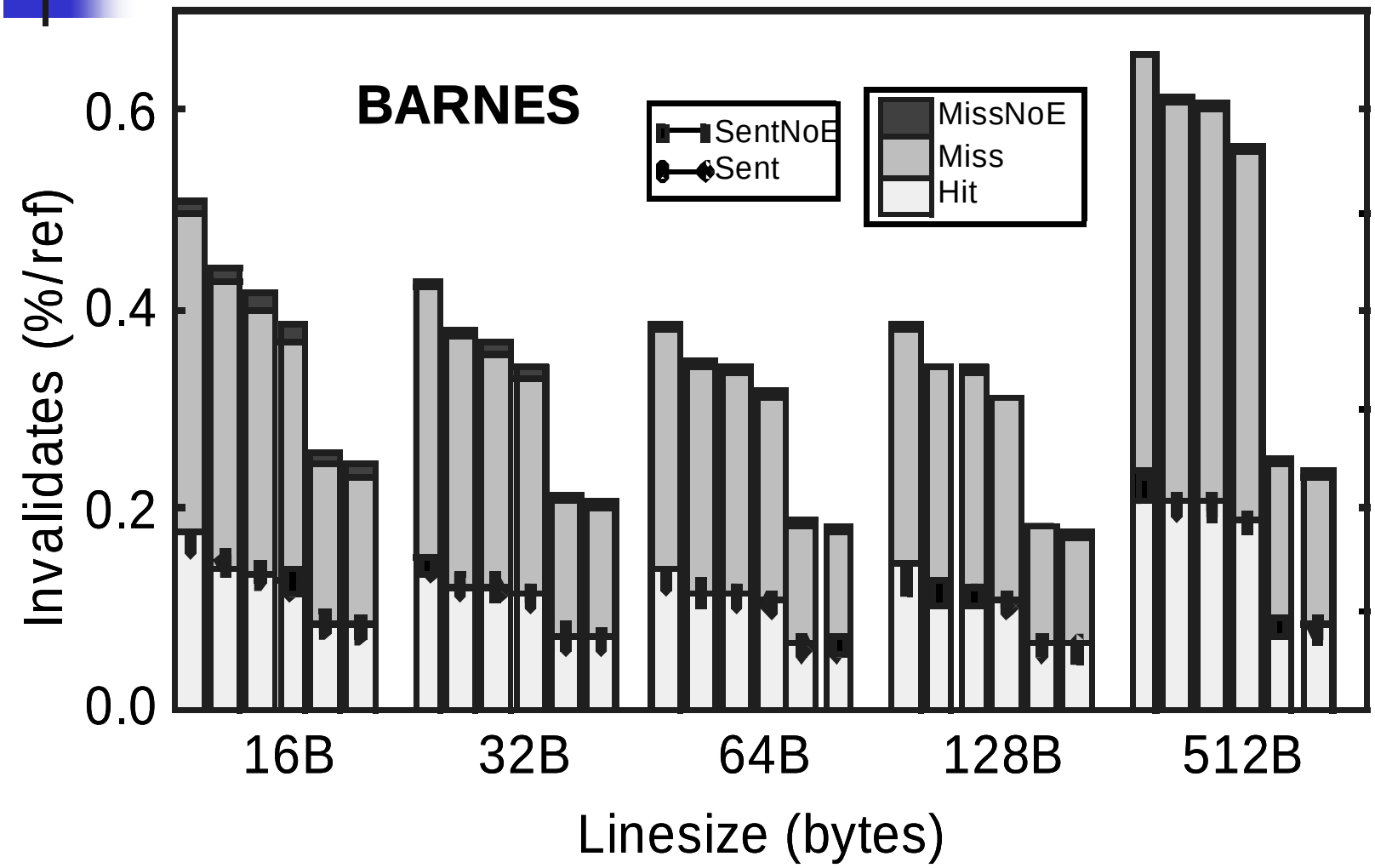
Previous Value Locality Works

- Lipasti, Shen: MICRO-1996, ASPLOS-1996
  - Load value prediction (VP), input register VP
- Mendelson, Gabbay: Technion TR-97
  - VP based on output register specifier
- Gonzalez, Gonzalez: PACT-1999
  - Improving branch prediction
- Calder et. al: ISCA-1999
  - Critical path optimizations
- Many Others

Multiprocessor Invalidates



Multiprocessor Invalidates



Multiprocessor Invalidates